WHAT IS CLAIMED IS:

1. A method used in a concurrent program analysis for detecting potential race conditions, such as data races, in a computer program, comprising:

receiving a source code of the computer program, the source code including an element annotated as either thread-local or thread-shared;

determining if the element is annotated as thread-shared or thread-local; and

verifying the validity of the thread-local annotation if the element is annotated as thread-local,

wherein an invalid thread-local annotation may cause a race condition.

2. The method of claim 1, wherein the computer program can spawn a plurality of threads that are capable of being executed concurrently, the method further comprising:

indicating a race condition warning or error if upon verifying the validity of the thread-local annotation of the element it is determined that the element is, in fact, visible from more than one, rather than one and only one, of the plurality of threads.

3. The method of claim 1 wherein, for any instance in which it is determined that the element is annotated as thread-shared, the method further comprises:

verifying that the element does not include a portion annotated as thread-local and/or a link to another element that is annotated as thread-local; and

indicating a race condition warning or error if the portion and/or the other element are annotated as thread-local.

4. The method of claim 1, wherein the element can be a global addressable resource and, if so, the method further comprises:

verifying that the element does not include a portion annotated as thread-local and/or a link to another element that is annotated as thread-local; and

indicating a race condition warning or error if the portion and/or the other element are annotated as thread-local.

5. The method of claim 3, wherein the element is a class structure, an object, a data structure or a record, the portion of which respectively being a class object, an attribute, a structure element, or a field.

6. The method of claim 1 wherein, for any instance in which it is determined that the element is annotated as thread-shared and includes a pointer or a reference to a different element, the method further comprises:

verifying that the different element is not annotated as thread-local; and

indicating a race condition warning or error if the different element is annotated as thread-local.

7. The method of claim 1, further comprising:

indicating a race condition warning or error if the element is thread-shared annotated and it is determined that the at least one portion of the element points to another element of the source code that is thread-local.

8. The method of claim 1, wherein the computer program can spawn a plurality of threads that are capable of being executed concurrently, and wherein verifying the validity of the thread-local annotation includes

checking whether at least one portion of the element, or another element pointed to by the element, is visible from more than one, rather than one and only one, of the plurality of threads, and

checking whether upon creation of a new thread of the plurality of threads the element is passed to the new thread, wherein a race condition warning or error is indicated if the element and/or the other element are annotated as thread-local but are visible from more than one, rather than one and only one, of the plurality of threads.

9. The method of claim 1, wherein the computer program can spawn a plurality of threads that are capable of being executed concurrently, and wherein verifying the validity of the thread-local annotation includes

checking, if the element is annotated as thread-shared, whether each portion of the element is also annotated as thread-shared,

checking, if the element is visible from more than one of the plurality of threads, whether the element is annotated as thread-shared, and

checking, if the element is passed into a new thread that is spawned from one of the plurality of threads, whether the element is annotated as thread-shared,

wherein an invalid thread-local annotation can prompt a warning indication.

10. The method of claim 1, further comprising:

checking whether a sub-element is derived from the element and, if so,

checking, if the element is annotated as thread-local, whether the sub-element is also annotated as thread-local,

checking, if the element is annotated as thread-shared,

whether the sub-element is also annotated as thread-shared, or

whether the sub-element is annotated as thread-local, and the sub-element does not override methods declared in the element and the element is not typecast to the sub-element.

11. The method of claim 10 wherein, for any instance in which it is determined that the sub-element is derived from the element, the method further comprises:

providing a race condition warning or error indication if the element is annotated as thread-local and the sub-element is not annotated as thread-local, or

if the element is annotated as thread-shared, the sub-element is annotated as thread-local, and either

the sub-element overrides methods declared in the element, or the element is typecast to the sub-element.

12. An apparatus for concurrent program analysis, comprising:

means for receiving source code of a computer program, the source code including an element annotated as either thread-local or thread-shared;

means for type checking the source code; and

means for checking annotations located either inside or in series with the type checking means, including

means for determining whether the element is annotated as thread-shared or thread-local; and

means for verifying the validity of the thread-local annotation if the element is annotated as thread-local,

wherein an invalid thread-local annotation may cause a race condition such as a data race.

13. The apparatus of claim 12, further comprising:

means for parsing the source code; and

means for creating from the source code an abstract syntax tree.

14. The apparatus of claim 12, wherein the computer program can spawn a plurality of threads that are capable of being executed concurrently, and wherein the means for checking annotations further includes

means for checking, if the element is annotated as thread-local, whether the element is visible from more than one of the plurality of threads, means for checking, if the element is annotated as thread-shared, whether each portion of the element is also annotated as thread-shared,

and

means for checking, if the element is passed into a new thread that is spawned from one of the plurality of threads, whether the element is annotated as thread-local,

wherein an invalid thread-local annotation can prompt the apparatus to provide a warning indication.

15. The apparatus of claim 12, wherein the means for checking annotations further includes

means for checking whether a sub-element is derived from the element and, if so,

means for checking, if the element is annotated as thread-local, whether the sub-element is also annotated as thread-local,

means for checking, if the element is annotated as thread-shared,

whether the sub-element is also annotated as thread-shared, or

whether the sub-element is annotated as thread-local, and the sub-element does not override methods declared in the element and the element is not typecast to the sub-element.

16. The method of claim 15, wherein, for any instance in which it is determined that the sub-element is derived from the element, the means for checking annotations further includes

means for providing a race condition warning or error indication

if the element is annotated as thread-local and the sub-element is not annotated as thread-local, or

if the element is annotated as thread-shared, the sub-element is annotated as thread-local, and either the sub-element overrides methods declared in the element, or the element is typecast to the sub-element.

17. A system for concurrent program analysis having a computer readable medium embodying program code for detecting potential race conditions, such as data races, in a computer program, including instructions for causing the system to:

receive a source code of the computer program, the source code including an element annotated as either thread-local or thread-shared;

determine if the element is annotated as thread-shared or thread-local; and

verify the validity of the thread-local annotation if the element is annotated as thread-local, wherein an invalid thread-local annotation may cause a race condition.